Secrets
ABAP
Apex
C
C++
CloudFormation
COBOL
C#
CSS
Flex
Go
HTML
Java
JavaScript
Kotlin
Kubernetes
Objective C
PHP
PL/I
PL/SQL
Python
RPG
Ruby
Scala
Swift
Terraform
Text
TypeScript
T-SQL
VB.NET
VB6
XML

Flex static code analysis

Unique rules to find Bugs, Security Hotspots, and Code Smells in your FLEX code

All rules 76 | Vulnerability 5 | Bug 9 | Security Hotspot 1 | Code Smell 61

Tags

Search by name...

Security.allowDomain(...) should only be used in a tightly focused manner
Vulnerability

The flash.system.Security.exactSettings property should never be set to false
Vulnerability

Dynamic classes should not be used
Code Smell

"LocalConnection" should be configured to narrowly specify the domains with which local connections to other Flex application are allowed
Vulnerability

"default" clauses should be first or last
Code Smell

Event types should be defined in metadata tags
Code Smell

Event names should not be hardcoded in event listeners
Code Smell

The special "star" type should not be used
Code Smell

Variables of the "Object" type should not be used
Code Smell

Methods should not be empty
Code Smell

Constant names should comply with a naming convention
Code Smell

All branches in a conditional structure should not have exactly the same implementation
Bug

Classes that extend "Event" should

"default" clauses should be first or last

Analyze your code

Code Smell | Critical

`switch` can contain a `default` clause for various reasons: to handle unexpected values, to show that all the cases were properly considered.

For readability purpose, to help a developer to quickly find the default behavior of a `switch` statement, it is recommended to put the `default` clause at the end of the `switch` statement. This rule raises an issue if the `default` clause is not the first or the last one of the `switch`'s cases.

Noncompliant Code Example

```
switch (param) {
  case 0:
    doSomething();
    break;
  default: // default clause should be the first or last
    error();
    break;
  case 1:
    doSomethingElse();
    break;
}
```

Compliant Solution

```
switch (param) {
  case 0:
    doSomething();
    break;
  case 1:
    doSomethingElse();
    break;
  default:
    error();
    break;
}
```

Available In:

sonarcloud | sonarqube

override "Event.clone()"
Bug

Constructors should not dispatch events

Bug

"ManagedEvents" tags should have companion "Event" tags

Bug

Objects should not be instantiated inside a loop

Code Smell

Two branches in a conditional structure should not have exactly the same implementation

Code Smell

Constructor bodies should be as lightweight as possible

Code Smell

Only "while", "do" and "for" statements should be labelled

Code Smell

Statements, operators and keywords specific to ActionScript 2 should not be used

Code Smell

"for" loop stop conditions should be invariant

Code Smell

Unused function parameters should be removed

Code Smell